PCT/GB 2004 / 0 0 1 2 1 8

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 1 June 2004

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

THE PATENT OFFICE
G
20 MAR 2003
LONDON

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference — SSA/P8682GB

2. Patent application number *(The Patent Office will fill in this part)* — 0306463.1 — 20 MAR 2003

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

STEELHEAD SYSTEMS LTD
MLFC Post Office Building
Floor 3
2 King Edward Street
London EC1A 1HQ
United Kingdom

Patents ADP number *(if you know it)* — 8597303000

If the applicant is a corporate body, give the country/state of its incorporation — England and Wales

4. Title of the invention — Improvements Relating to Communications Data Management

5. Name of your agent *(if you have one)* — David Keltie Associates

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

Fleet Place House
2 Fleet Place
London EC4M 7ET
United Kingdom

Patents ADP number *(if you know it)* — 040145020006

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

YES

Patents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	
Description	25
Claim *(s)*	
Abstract	
Drawing *(s)*	15 + 15

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	

11. I/We request the grant of a patent on the basis of this application.

Signature: David Keltie Associates — Date: 20 March 2003

David Keltie Associates

12. Name and daytime telephone number of person to contact in the United Kingdom: Dr. Shakeel Ahmad, 0020 7329 8888

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.

d) If you have answered 'Yes' Patents Form 7/77 will need to be filed.

e) Once you have filled in the form you must remember to sign and date it.

f) For details of the fee and ways to pay please contact the Patent Office.

Improvements Relating to Communications Data Management

Field of the Invention

The present invention concerns improvements relating to communications data management, and more particularly, though not exclusively, to an automated method of organising a user's inbox to sort incoming e-mail (electronic mail) messages. The present invention also has application to the remote control and updating of applications using e-mails and the formation of secure e-mail consortia.

Background of the Invention

Ever since the emergence of the Internet as a viable and reliable communications network, text-based communications via the Internet, namely e-mail has proliferated. Many communications which would previously have been effected by facsimile or telephone are instead now handled by e-mail. The reason for such a change in the manner of communication is due to several advantageous factors. Unlike conventional mail, an e-mail can be delivered to a recipient in a matter of minutes regardless of the recipient's location. Additional information can also be sent with the e-mail such as an electronic file or a hyperlink such that the need for the sending of storage media can be obviated. Furthermore, the cost of sending an e-mail is a fraction of that of conventional mail or even a facsimile.

Various e-mail packages such as Microsoft Outlook® Lotus Notes® and Novell Groupwise® exist for handling the creation, sending and reading of received e-mails. Each of these packages creates an inbox on the recipient's computer where e-mails are initially stored that have been retrieved from the recipient's e-mail address on their respective e-mail server. All new messages are typically highlighted such that all unread e-mails are brought to the recipient's attention. Thereafter, the recipient can read the e-mails and determine the category to which the information contained in the e-mail belongs. Subsequently, the recipient can decide where to store (file) the e-mail in their e-mail folder structure or whether it is to be deleted.

With the prolific increase in the number of e-mail communications that recipients are now receiving, managing the recipient's inbox to ensure that each received e-mail is properly filed has become an onerous task. There is a significant problem with manual sorting and filing and attempts have been made (see for example US 6,216,165) to address this by the introduction of what have been termed 'automated' e-mail filing methods. These methods involve the user having to manually set up message-specific rules, which act on incoming e-mail to recognise and thereafter act on received e-mails. This is achieved by searching the subject field of an e-mail for predetermined words or recognising a predetermined sender's address in the header field. Sometimes the required programming needs to access an external database to achieve its desired effect.

However, such methods are difficult to set up and maintain placing a significant burden on the receiver. Typically, users tend to set up such methods initially but not maintain them due to this burden. In any case, such systems and methods can at best only be partially successful because e-mails from new senders regarding new subjects or in a 'free text' format cannot, by definition, be handled by such systems.

Another problem with existing manual and automated methods is that old e-mails (i.e. those outdated by similar e-mails with newer content) persist and require deletion. The problems caused by such e-mails is particularly evident in the case where there are a stream of e-mail messages transmitted to a recipient regarding a constantly changing parameter, such as a stock market quote. Here the recipient's inbox can quickly become clogged with e-mails to be read as well as outdated information.

Some of these problems amongst others have been considered in International Patent Application WO 01/76264A and this has lead to the creation of a centralised web mail service where the web based communications contain an XML message. This solution is complex and relies on a central organising server to arrange data destined for a recipient. Also, this solution requires a person wishing to participate to register with the central server and to send messages using Internet-based communications. The use of web communications means that messages are susceptible to screening and blocking by firewalls. This is because the content of such Internet communications can be analysed and if found to be prohibited, can be filtered preventing the user receiving the message. Furthermore, this prior art system implements a pull model where information has to be pulled from a web site. This is not ideal in that it makes the recipient have to implement more procedures (such as the setting up of an alert channel) to get the data than in a push model.

It is desired to overcome or at least substantially reduce the above-described problems by the provision of an improved method of handling received e-mails.

Summary of the Present Invention

According to one aspect of the present invention, there is provided a method of filing a received e-mail message, the method comprising: reading a text-based data structure within the received e-mail message; comparing the data structure to a plurality of pre-stored e-mail data structures; and storing the received e-mail message in a folder to which the data structure corresponds.

The e-mail is therefore categorised by way of its data structure and is filed truly automatically without requiring any registration or predetermined configuration of the sender's machine. This advantageously keeps inbox clutter to a minimum with there being no requirement for the recipient to configure or arrange his machine to accept these new types of e-mails.

The use of a text-based data structure within an e-mail, means that there are no problems with firewalls as the text within an e-mail does not get parsed for determining whether to block a received communication. In this way the present invention does not suffer from the problems described in relation to web-based communications.

The present invention enables data to go straight into a recipients inbox which advantageously operates as a push model rather than a pull model. In this way the recipient does not have to set up of carry out any special procedures to view the data.

The present invention, using an e-mail message:

a) allows structured data to be communicated via e-mail whilst retaining its structured form and to be viewed by the recipient in its structured form.

b) creates folders specifically designed to hold a particular data structure without any intervention by the recipient.
c) files similarly structured messages automatically and allow full sorting / searching capability on any field defined in the structure.
d) allows multiple records – i.e. a multiple row dataset - to be packed up and sent in one standard e-mail message and be reconstructed into individual records by the recipient's computer.
e) allows data of a changing nature to be sent via e-mail with the recipient always seeing the latest view only.
f) allows structured data contained within e-mails to be automatically written to a database file external to the e-mail application.

The present invention has the following specific advantages/features:

a) Structured data can be sent and replied to via e-mail codified within the message itself, without the need for attachments / external programs (e.g. spreadsheets) to view the data. Sending e-mails without attachments means; no firewall problems,no virus risk and no action required by user (e.g. opening the attachment).
b) New data structures can be sent to recipients and automatically folders are created and e-mails filed thereby reducing filing or rule creation effort by the end user and creating bulletin-board like functionality within e-mail clients.
c) Similar-structured data from multiple recipients can be co-mingled and combined into one view and fully sortable instantly. Previously structured data sent as attachments to e-mail had to be combined by the recipient into one file to allow data sorting across all records.
d) A user can send multiple records of a database or send multiple messages to the same recipient within just one standard e-mail message with no attachments. This saves in-box clutter and reduces network traffic and processing. Once received the single e-mail reconstructs into individual records allowing sorting, sifting and other data manipulation by the recipient.
e) The recipient does not need to discard old messages as this clean-up is done by the sender. Using the applicant's program (Steelhead Application), the sender has control over the messages as displayed at the recipient's computer and can delete out of date records remotely, or update information where it has changed.
f) No new data viewer or parsing application is needed by the recipient, and records received are automatically accessible by other applications in the recipient's environment, thereby allowing corporate systems to communicate without opening up incremental firewall holes as the e-mail server connections are used and only text information is being transferred.

Further advantages of the different aspects of the present invention are described later.

In summary, different aspects of the present invention described in the specific description address at least the following seven problems experienced by e-mail users prior to the invention:

1. e-mail often arrives in a "free text" format, and as such similar e-mails (i.e. those containing similar data) from different senders are not co-mingled without cut/paste effort on the part of the recipient;

2. messages are not filed on receipt without setting up pre-defined in-box rules;
3. old e-mails (i.e. those outdated by similar e-mails with newer content) persist and require deletion;
4. groups of data contributors (referred to as consortia) have no "enforcer" of a data structure when sending data the same recipient(s) by e-mail in supposedly the same format. This allows errors in data structure on the part of one or more senders to jeopardise the combined recordset the receiver is trying to combine from the multiple e-mails;
5. requests for information sent to multiple recipients asking for feedback in a pre-defined format are not automatically co-mingled upon receipt, that is the data in the reply e-mails are not co-mingled as returns come in without some pre-programming effort typically requiring an external database;
6. creating an encrypted environment requires the sender and recipient to co-ordinate prior to the sending the e-mail which is not always convenient. For example, the recipient prior to this invention would be required expressly to make their public key available to the sender; and
7. cross referencing of information between e-mail folders was not previously possible as data structures can be defined to be extensible with table database style table joins possible using the structured nature of the data.

Brief Description of the Drawings

Figure 1 is a schematic block diagram showing a communications system within which an apparatus according to a first embodiment of the present invention is implemented;

Figure 2 is a flow diagram showing the operation of the apparatus of Figure 1 in implementing the first embodiment of the present invention;

Figure 3 is a screen shot representation of an inbox generated by the apparatus of Figure 1;

Figure 4 are examples of the XML code structure of a received e-mail according to the first embodiment;

Figure 5 is a schematic block diagram showing the way in which the apparatus of the present invention interacts with senders and other applications within the recipient's computer;

Figure 6 is a schematic block diagram showing a communications system within which an apparatus according to a second embodiment of the present invention is implemented;

Figure 7 is a schematic block diagram showing the Schema manager of Figure 6 in greater detail;

Figure 8 is a flow diagram showing the operation of the Schema manager of Figure 6 in implementing the second embodiment of the present invention;

Figure 9 is a flow diagram showing the operation of the whole apparatus of Figure 6 in implementing the second embodiment of the present invention;

Figure 10 is a continuation of Figure 9;

Figure 11 is a screen shot representation of a spreadsheet type grid (which replaces the normal new mail window) for input of data generated by the apparatus of Figure 6;

Figure 12 is a screen shot representation of a series of drop down boxes used for defining a unique key that is a combination of the fields of the schema, the sender's e-mail address, read receipt recipient and the subject / schema title for example;

Figure 13 is a screen shot representation of a field chooser type view option box which can be used select columns that the sender has sent which the recipient is interested in viewing;

Figure 14 is a screen shot representation of a spreadsheet plug-in used to generate new e-mails directly from the spreadsheet that fit the schema requirements;

Figures 15 to 21 are respective flow diagrams showing the different functional components of the embodiment of the present invention.

Detailed Description of Preferred Embodiments of the Present Invention

Referring to Figure 1 a system embodying the present invention is shown. Figure 2 shows a flow chart of the steps taken in the creation, sending and filing of a structured e-mail (sometimes referred to as Clovismail).

The apparatus embodied in a software application, seeks to allow the user to read, write, send and receive XML messages passed over e-mail which, on receipt, dynamically constructs a view onto the data at the recipient's computer. This view allows the content of the message to be manipulated based on any of the fields defined by the sender rather than the traditional e-mail practice of sorting by say Sender, Subject, Time Received etc. The application does this by sending both the XML schema describing the data and the associated data fitting the schema structure in a tagged XML format (namely as text characters) in the body of the application.

Using this concept, multiple senders can send information in identical data structures to one recipient and this content is automatically co-mingled for the recipient as if it were one original recordset irrespective of the initial multiple sources of information. In this way problems requiring multiple senders to contribute previously unstructured information to one end user in a co-mingled and data structured way are solved within an e-mail context.

This concept is enhanced further as a single recipient may of course receive varying data structures from different senders. As such the application automatically creates a folder containing for each distinct XML schema / recordset. Incoming messages are then filed automatically into the correct folder using the following logic:

- if the data structure is recognised from an earlier message, the e-mails are filed together;

- if not a new folder is set up and the new data structure and associated data filed accordingly

This solves the need for the recipient to file incoming messages and helps de-clutter the recipients inbox. As similarly structured content is delivered to the user over time, a database key system is definable by the sender to specify that new e-mails replace old e-mails (again filed in the correct older) to ensure that e-mail content is up to date and old e-mails are deleted automatically without recipient intervention. As such, a delete and insert operation effects an update operation on the data. In this way the sender can control the view (both structure and timely content) that the recipient has on data the sender wishes to send.

The application therefore leverages the ease of e-mail and combines with database style functionality without any database knowledge required on the part of the recipient.

A consequence of this data structure technique is that multiple folders will appear on the recipients machine as varying schema are sent over time.

Receiving a structured e-mail

As described above, a user receiving a structured e-mail will have the data content automatically filed based on whether the schema definition matches a structure received in a prior message (see Figure 2). On clicking on the appropriate folder, contents of any data within that schema are shown. The screen-shot set out in Figure 3 highlights a folder called "Trading Axes" (a bond markets term to highlight special offers) with 8 fields of data. The folder shown in Figure 3 was created by the sender not the receiver. The sent message took the format shown in Figure 4.

Applications

Applications of the different aspects of the present invention as listed under the lettering used above are:

a) The Steelhead Application turns standard e-mail messaging into a connectivity system for structured data communication. As such the following structured data applications could evolve in areas that are enhanced by structured information (not exclusive list):
   - Pricing (e.g.: Bond Prices, Stock Prices, Commodities, Other Financial Instruments, Catalogues, Gambling odds, Products and Service Prices and Offers)
   - Listings (e.g.: Property, Entertainment, Classifieds, Timetables, Directories)
   - News (e.g.: Business News, Weather and Sports Information, Surveys, Research, Reference)
   - EDI functions (e.g.; Requests for Proposals (RFP), Bids, Inventory, Orders, Invoices, Transactions)
   - Management information (e.g.: Budgets, Project updates, Client and sales information, Policies, Surveys, Market Research)
   - Group Communication (e.g. creating distributed bulletin board like folders around certain topics)

b) Same as a)

c) Same as a)

d) Same as a) and especially for bulk data requirements that are likely to be found in Pricing and Listings applications where there is information on hundreds or thousands of records to be delivered to the recipient.

e) Same as a) and d) and especially for frequently changing data requirements that are likely to be found in Pricing applications particularly in Financial markets and News applications where the same record has fields that change regularly so require modification rather than additional data addition.

f) Same as a) and especially EDI functions where another application will react to the data sent - for instance RFP data will result in a call to an automated Bid engine or invoice data will be automatically picked up and entered into an accounting system.

Methodology

a) The Steelhead Application allows structured data to be sent via e-mail whilst retaining its structured form through the creation and recognition of "Structured E-mails": This is performed as follows:

i) Sender has or constructs information they wish to communicate in a structured format, typically taking the characteristics of any 2 dimensional table or list. For example the following table of data:

A	B	C	D	E			
R	O	W	1				
R	O	W	2				

ii) A component (either in senders e-mail, spreadsheet or some other application) formats, and may encode, the structured data set as a text string that can be contained in the e-mail message.

Pseudo codification:

Schema name = "Gridder"

Data Structure:
4 fields
1st Field named "A"
2nd Field named "B"
3rd Field named "C"
4th Field named "D"

1st Field data type = "text"
2nd Field data type = "text"
3rd Field data type = "text"
4th Field data type = "numeric"

Record 1:
Field "A" = "R"
Field "B" = "O"

Field "C" = "W"
Field "D" = "1"
...
Record 2:
Field "A" = "R"
Field "B" = "O"
Field "C" = "W"
Field "D" = "2"
...etc

iii) E-mail routing information is added so that the message behaves and is treated as a standard e-mail, and the e-mail is sent via standard e-mail paths.

iv) A component at the recipient's end recognises the e-mail as a Structured E-mail and unwraps and restores the structured format of the data from within the E-mail. The view onto the data for the recipient is similar to the traditional "inbox", except that it will contain the original tabular structure that the sender commenced with. No additional viewer is therefore needed. Example view for recipient:

A	B	C	D	E			
R	O	W	1				
R	O	W	2				

b) Folders are created by the Application on the recipient's PC using the following methodology:

i. E-mail is detected by component installed on recipients computer as a Structured E-mail by scanning the text string to ensure it adheres to the Application's requirements of a schema ID, data structure definition and data.

ii. The Schema name is read. In the following example = "Gridder"

Schema name =Gridder

A	B	C	D	E			
R	O	W	1				
R	O	W	2				

Pseudo codification:

Schema name = "Gridder"

Data Structure:
4 fields
1st Field named "A"
2nd Field named "B"
etc.....

iii. If a folder with the Schema name is present the e-mail is filed within that folder. In this example if a folder with Schema name Gridder was present, the e-mail would be filed within that folder.

iv. If no folder exists for the specific Schema Name then a new folder will be created in preparation to hold the data. In this example the folder will be named "Gridder"

v. In all cases the folder is now viewable on the recipient's PC as part of the recipient's folder tree view:

- Inbox
- Sent Items
- existing folder 1
- existing folder 2
- Gridder

c) All new messages arriving at the recipient, regardless of sender, that are Structured E-mails and that have the same schema, are automatically filed in the same unique folder on the recipient's PC. In the above example all Structured E-mails from any sender with a Schema name Gridder would be filed in the Gridder folder. The key non-obvious methodology advance is different senders can all automatically file their messages into the same unique folder on the recipients PC.

d) Now that individual folders for different data structures have been created, this allows similarly structured data to be grouped together. The key non-obvious methodology advance here is that each record of data becomes a new row in the users folder even though only one original message was sent.

Therefore, continuing the above example, when the user clicks on the "Gridder" folder to view its content they see:

Gridder

A	B	C	D
R	O	W	1
R	O	W	2

This view is constructed even though only one original e-mail was sent. A user is unaware that the data came in one e-mail message not two as the records in the data are now fully independent. As such a user can then sort data on any data item (1st click ascends, 2nd click reverses i.e. descends). In this example a user clicking on the fourth field "D" could view data as:

(ascending sort on field "D")

A	B	C	D
R	O	W	1
R	O	W	2

Or

(descending sort on field "D"

A	B	C	D
R	O	W	2
R	O	W	1

e) The recipient does not need to clean up their folders as whilst new data is inserted as described above, the sender can also delete and update previously sent data. This methodology for this works as follows:

i. The sender can add additional attributes to their message to allow previously-sent e-mails to be deleted or updated similar to the "DELETE" and "UPDATE" operations in normal SQL

ii. To update, the sender configures the e-mail appropriately through some GUI and then specifies which fields form the matching criteria. E.g. a user may specify that "Field C" and "Field D" are relevant in this matching criteria and therefore if any of the records sent with this 2nd updating message contain information in "Field C" and "Field D" that matches any records in the recipients inbox from the same sender, then the previous records will be deleted and the next content inserted. This is equivalent to a database SQL command of UPDATE WHERE "Field C" In ([list of field C values sent in this message], ...) AND Field D In(...,..., etc)

iii. To delete, the sender configures a message appropriately and similar logic is applied. This is equivalent to SQL-style "DELETE..WHERE" operation on the recipients folder.

Pseudo codification for an update:

Schema name = Gridder
Record 1:
Field A = "G"
Field B = "H"
Field C = "W"
Field D = "1"
...
Record 2:
Field A = "I"
Field B = "J"
Field C = "W"
Field D = "3
...
Field match: Field C, Field D

Example:
Therefore if the users inbox previously looked like this:

Schema name=Gridder

A	B	C	D				
R	O	W	1				
R	O	W	2				

Normally the new e-mail would be filed as 2 new rows to generate a 4 row inbox looking like:

Schema name=Gridder

A	B	C	D				
R	O	W	1				
R	O	W	2				
G	H	W	1				
I	J	W	3				

However as the Field match criteria were set to "Field C" and "Field D", then if any records match either "W,1" or "W,3" in fields Field C and Field D respectively, then the previous records would be deleted and the new records inserted, in effect an update. That is, the new inbox would look like:

Schema name=Gridder

A	B	C	D				
G	H	W	1				
R	O	W	2				
I	J	W	3				

To explain this, the 1st record in the update message (G,H,W,1) matched the original 1st record in the recipients inbox (R,O,W,1) in the last 2 fields (Field C and Field D) – therefore R,O,W,1 was deleted and G,H,W,1 was inserted. The 2nd record in the update message (I,J,W,3) did not match any content in the users inbox and therefore was appended as normal.

iv. Similarly coding a Structured E-mail with expiry times will also allow messages to self-delete after a certain time period, thereby cleaning the content of the folder automatically for the recipient

f) When the recipient's PC receives a Structured E-mail, the individual data records are simultaneously:

i. viewed within the recipient's e-mail application as record tables, and
ii. exported as CSV, Excel, XML or any other standard data structure language into a separate file on the users PC outside of their e-mail application. As the raw data records are stored externally to the e-mail application, these can be simultaneously accessed by other applications within the recipient's organisation. A basic flow diagram in relation to a firewall is shown in Figure 5.

Schema control

This aspect of the present invention is described with reference to Figures 6 to 14 of the accompanying drawings.

To control schema implies the following:

1) schemas with errors are not set up as new folders, but returned to sender as bad data
2) non consortia members cannot send data to folders they do not participate in
3) secure transmission of information
4) consortia members cannot read each others content (unless desired!)
5) data hacked on internet cannot be read
6) schema structure updates are managed seamlessly across large numbers of users
7) sophisticated structures (dynamic links between folders) can be created e.g. Bond.Hub axes linked by ticker to Bond.Hub research (Bond.Hub is an existing industry consortia set up between 8 dealers (ML, CSFB, GS, JPM, Lehman, MSDW, SSB, UBS). All dealers have integrated their websites (including single logon and password) with a view to providing a single website that combines their research articles and also their trading axes. Axes are typically like "today's special offers" where a dealer wants to promote a certain bond because they feel it is worth trading and they are offering a good price. Because all dealers are promoting their best prices, clients want commingled lists of bonds across all dealers.)

Although a product allowing the service to be used without a schema manager is possible, the following downsides are likely:

1) errors can occur (resulting in multiple folders even though data should be commingled)
2) non-consortia members (those without the official schema number and an encryption key) can participate in these folders if they know the basic data structure
3) data transmission is not encrypted
4) folder chaos

How a schema manager service of the present invention works:

The following process mitigates the disadvantages described above.

1) The schema manager runs a web tool to allow easy management and registration of schema
2) Consortia log-on to register & edit schemas across multiple end users
3) Process eliminates need to upload client lists to central schema manager

The schema manager web tool provides:

1) an official schema number for each registered schema

2) a public key for encryption of e-mails (one key per sender)

3) a private key for decryption of e-mails (sent on request by the schema manager to each recipient the first time a new sender sends a new schema)

4) a control report of which users received which content

Sending a new schema

Process Flow

1) Different senders encrypt their own content using their public keys received from the schema manager

2) The first time the recipient receives this specific schema (via an unencrypted schema ID) from a specific sender, the component automatically contacts the schema manager for the private decryption key to decode full message

3) Subsequent messages in this schema from this sender are decrypted automatically

Quality Control of Messages

Process Flow

1) E-mails are sent in the agreed format

2) For registered schema component logic differs to free version: any deviations from schema (e.g. errors) are not filed under a new folder, but instead returned to sender with a message describing error (i.e. which part of message did not match)

Updating Folder Structures

Process Flow

1) As schema is updated on Schema Manager web site, new ID is created for schema

2) The first time the recipient receives this new schema ID (as yet unrecognised) the component automatically contacts the schema manager for the private decryption key to decode full message (per normal)

3) On receipt of the key, the component is told that in this case a folder update, not new folder is required. Component makes changes automatically

Complex Structures

The services of the schema manager can facilitate the setting up of complex structures:

1) Web tool allows simple way of linking data between registered folders:
   e.g. consider a price folder and research folder with a data link based on the Ticker field present in both folders
   Bond prices folder (Ticker, Coupon, Maturity etc)
   Research (Headline, Ticker, Sector, Author)

2) e-mail component will then allow certain enhanced actions (eg right clicking a ticker in the Bond Prices folder pulls up a menu of related research headlines (same Ticker) drawn from the Research folder

Functional Overview:

This section describes the core functionality in the product as is a simplified version of the technical specification document used.

1. The XML schema and recordset is sent in the body of the e-mail
2. The sender's component encodes the XML
3. The recipient's component decodes the XML and converts the message into a manipulatable recordset.
4. The recipients component recognises distinct schema contained in the e-mail body and arranges each schema in a new sub-folder where each individual record adhering to that schema is listed as a distinct "message" that can be read
5. Distinct schema are defined as:
   a) does not have the same subject title as any other message AND
   b) does not have an exactly matching data structure
6. Looking like an e-mails in box, the normal column headings of From, Subject, Received date are replaced the column headings described in the schema

   (in the example shown in Figure 3, the column headings are: Ticker, Coupon, Maturity, Buyer/Seller, Size(MM), Price, Spread, Benchmark)
7. If a message (containing one or multiple records) is sent that matches an existing schema / folder, then the data is filed in that folder. If the mail does not match, then a new folder with a data structure as defined in the message is created.
8. Records are sortable / viewable just like e-mail now – i.e. by clicking on the column titles the records are re-ordered by that field.
9. Records (each record has its schema built into the message) can be forwarded to other users where on receipt the record will cause either a new folder to be created or will group the record with other records if the schema is already in user on the recipients machine
10. Users can originate new messages and schema by completing a spreadsheet type grid (which replaces the normal new mail window and is generated automatically from the Outlook plug-in) shown in Figure 11. Fields across top, records going down, on pressing send an XML message is generated. E-mails are created with the body of text being replaced by commands that can generate a grid which a user can complete with data. An options part of the message allows the user to specify a unique key. Also messages can be created from Excel spreadsheets, with a new message being called up based on a selected set of cells, the options part of the message allowing a user to specify a unique key.
11. The super-schema is required to control overall features of communication with certain "super-parameters". This is controlled from an "Options" tab in the new mail section. Super-parameters are shown below
    a) A new header entry that describes message as a component message, and hence deposits the message in a folder not the regular e-mail.

b) Each record has an "Expires by" feature (defaulted to 24 hours) to automatically delete old records and "self-clean" the folders

c) Each record has a specified "Read receipt recipient" (e-mail address) to send the read receipt to a user who is not the sender (e.g. if the original e-mail was computer generated, but a salesperson would like the read receipt). Read receipts are themselves in XML format and follow the schema of the original message (but titled as ReadReceipts: XXX – where "XXX" is the original name of the schema sent). The default for this field is the sender.

d) The overall schema has a "Schema Persists (Yes/No)" feature to determine whether a schema without any records (say, if have all been deleted after expiring after 24 hours) still exists as an empty folder or is automatically cleared

e) Allow forwarding (Yes / No) by record will allow users to specify certain items they do not want forwarded to other users by the first recipient.

f) Each schema when defined can be set to have a unique key that is a combination of the fields of the schema, the sender's e-mail address, read receipt recipient and the subject / schema title. This is shown in Figure 12 by the series of drop down boxes (up to 10 fields can be combined) to form a unique key. The check box "Erase previous messages with unique key" then allows the sender to delete previous e-mails they have sent, thereby allowing updates via a Delete & Insert operation

12. Any message sent is stored in a folder called "Previous schema" that allows the user to create new messages in a set schema format by calling on the schema of a previous record and not generating the data structure / schema from scratch for subsequent messages.

13. Within any given schema, the complete recordset (list of messages in that schema) can be exported to excel or comma delimited file

14. Messages can have multiple recipients

15. Normal mouse controls will be available on messages

16. Messages when clicked open up as would a normal regular e-mail – this is useful in the case of large records where in the folder view, only a truncated view of a field's contents are show (e.g. "Lower interest rates will..." could be expanded when the record is opened to view the whole text.)

17. In the users Clovismail - inbox / folder view, columns can be re-ordered using drag & drop and a field chooser type view (shown in Figure 13) to add or hide columns that the sender has sent, but the recipient is not interested in. Therefore this allows customisation of the users view of the data that persists for new records matching that schema sent at a later date. The field chooser is shown in Figure 13 – in this case the user is dragging in the extra field "Benchmark ISIN"

18. The body of each mail sent should start with some untagged free text message saying "To decode this message download this component from www.....". This is useful in the case of sending the content to users without the component installed as it informs them how to get the component as the message is deposited in their inbox. If a user DOES have the component, as the initial message is untagged, it is simply ignored and the message is filed as described above.

19. Ability to copy an Excel list into the New Mail writer detailed in point 10 above and Figure 10. The list would need to be in standard grid format (for example 5 columns x 8 rows) with no blank lines etc.

20. A spreadsheet plug-in to generate new e-mails DIRECTLY FROM THE SPREADSHEET that fit the schema requirements and can be converted to textual characters for embedding within the e-mail itself, rather than the known method of creating a new mail with the excel file as an attachment. An example diagram is shown in Figure 14.

21. Read receipts for messages should be sent at the point of opening the folder in which the messages reside – not later when the actual individual records themselves are opened. For example if a user had a folder/schema called "Trading Axes" with 5 new records sent to it. The folder would display in bold: **Trading Axes (5)**. On clicking the folder to view any of the items – all records should be marked as read and one read receipt per key omitting the user-defined part of the key (see options section of a new mail message, the key is formed by combining: sender's e-mail address, read receipt recipient and the subject / schema title and any user specified key records) is generated at this point. It is assumed that opening a folder equates to reading ALL content of ALL messages. This is because individual records are not likely to be opened in many cases – only read from the folder view.

Figures 15 to 21 show the different process flows which are carried out during use of the present invention to assist in better understanding of the present invention.

The present invention has a further aspect in that it can be used for implementing remote control via e-mail messaging. This is described further below.

Remote Control via E-mail

Aims

Using an e-mail with no attachment:

a) To update a sender-defined database on the recipient's computer.
b) To update the functional capability of the applicant's program used with the present invention.
c) To update the executable code of the applicant's program.
d) To issue commands to the applicant's program.
e) To issue commands indirectly to other programs.

Advantages

Absence of attachment means; no firewall problems, no virus risk and no action required by user (e.g. opening the attachment)

a) By using standard e-mails single or multiple senders can use e-mail to keep a database up-to-date on a recipient's computer without the recipient needing to set up rules and/or define the database. This would normally need a separate file attached to the e-mail or would rely on a recipient-defined database where the recipient has set up rules to run a parsing program to reconstruct structured data from the unstructured text of an e-mail.
b) Would normally require a user-activated download and reinstallation – However in the present invention updates received by e-mail and are transparent to the user.

Each user may have a different and changing feature profile, while having the same version of the program.

c) Would normally require a user-activated download and reinstallation – However in the present invention updates received by e-mail and are transparent to the user.

d) Would normally need a direct network link but in the present invention use existing e-mail links.

e) Would normally need a direct network link but in the present invention use existing e-mail links.

Applications

a) The updated database could be any data structure or content defined by the sender(s). The use of this is as broad as is the use of data. Some examples (not exclusive list):
   - Pricing (e.g.: different Brokers updating bond prices and availability for clients)
   - Listings (e.g.: members updating membership directories with various associations)
   - News (e.g.: Legislators updating policies, codes and rules with relevant lawyers)
   - EDI functions (e.g.; manufacturers updating pricing and availability information with retailers)
   - Management information (e.g.: Sales people updating pipeline and client status for management teams)

b) To immediately upgrade the program via e-mail, whenever commercial or other considerations dictate either 1) on a user by user basis or 2) to all users, and either 1) on a feature by feature basis, or 2) a complete upgrade. Examples of use (not exclusive list):
   - To respond speedily to a user's request for a "new" feature
   - To turn on and off selected features based on client payments
   - To increase gradually the complexity of the product to "educate" the user
   - To reward users with new resources or credit (e.g. levels in a Game, time to run the program, quantity of download allowed or amount of processing allowed)

c) As b) above to update the whole executable code, although not usually practical for a feature by feature or user by user basis.

d) To command the applicant's program to run certain routines. Examples of use: (not exclusive list):
   - Remote control of authorisation and licensing,
   - Log-file and error-report-collection

e) Control of other applications by e-mail, by authorised senders and by their clients. Examples of use : (not exclusive list):
   - Help-desk trouble-shooting
   - Distributed idle-time research programs
   - Adding resources or credit to maintain other applications running
   - Remote control by client to manage other virtual or physical applications such as burglar-alarms, appliances, answer-machines, lights (assuming relevant hardware present)

Methodology

Using Structured E-mails or unstructured e-mails, with recognisable characteristics

a) A "Program-Flow" file (a text file within the Steelhead Application on the recipient's PC) instructs the Steelhead Application (see Figure 5) on how to deal with data contained in e-mails. Consequently e-mails containing data are identified by the Steelhead Application and the data contained within the e-mails are automatically exported out of the recipient's e-mail application as CSV, Excel, XML or any other standard data structure language into a separate database file on the recipient's PC. Similarly, new data contained within new e-mails can automatically update or replace existing data in such database files on the recipient's PC (Figure 5).

b) Certain features (e.g. menu items) are designed and coded in the originally distributed program executable, but only operate if appropriate flags appear in a Program-Flow file, which can be updated via an e-mail. The consequence would be that from receipt of the upgrading e-mail, flags within the Program-Flow file would change and the user would see, for example, new menu items, giving access to new commands (e.g. filtering, sorting, archiving, data exporting ...).

c) An e-mail will be defined as a 'Command' E-mail and a command line (encrypted) will be contained in the body of the e-mail. The new version of the program is encrypted into alphanumeric format, and sent as a Command e-mail. This is decoded on receipt and saved to disk, to replace the previous version of the program, which will run the next time the program is started.

d) A command line (encrypted) will be contained in the body of the e-mail. The Steelhead Application will identify the command line as intended for the Steelhead Application and will run that command line on receipt of the e-mail.

e) As above, an e-mail will be defined as a Command E-mail and the command line (encrypted), and the program to which it pertains, will be contained in the body of the E-mail. The Steelhead Application will identify the command line as intended for another program and will pass that command line on to the appropriate program.

Detail

a) : **Remote updating of database via e-mail.**

The Program-Flow File describes how the data contained within an e-mail is treated. The following pseudo-code illustrates the process.

Pseudo Code

```
..
CheckInBox
For Each Message
  If Message.Type = "Data"
    NewData = Decrypt (Message.Data)
    ParseData NewData
    Save NewData,NewData.DataName
  Else
    ..
  End if
  Else
    ..
  End if
Next
```

Example:

Data exists on "Department Manager's" PC in a file called "Salaries" to help Department Manager track the details of temporary staff hired by different agencies around the world. For example the original file may be as follows:

Department Manager PC / Filename = Salaries (original)

```
Sender,Name,Title,Age,Address,PostCode,Salary
HR(NY),J. Doe, Mr,33,1 Anywhere,10005,28000
HR (Boston),E. Kennedy,Chappaquidick Drive,12345,350000
HR(London),A. Smith,Ms,25,1 Armageddon,SN9 3AA,22000
```

The file is to be updated by HR Department in an employment agency in London "HR London" to inform the Department Manager of both; an increase in salary for Ms A. Smith and the fact of a new hire of Mr. J. Smith

New Data from sender "HR London":

Name	Title	Age	Address	PostCode	Salary
A. Smith	Ms	25	1 Armageddon	SN9 3AA	29000
J. Smith	Mr	45	45 Jones Drive	SN1 3AA	18000

This data is embedded by HR London in a structured e-mail to be sent to Department Manager.

E-mail

To: Department.Manager@client.com
From: HR.department.London@agency.com
Subject: Salary change and new hire

This e-mail contains data for your Steelhead software. If you can see this message, please goto www.steelhead.com/downloads to download the software.

```
<steelhead>
  <senderlicence>2h8d739487f9s8377h4hnnzx</senderlicence>
  <mailtype>data</mailtype>
  <uniquerecordID>Name,Title</uniquerecordID>
  <data>
    <dataname>Salaries</dataname>
    <row>
      <action>update</action>
      <Name>A. Smith</Name>
      <Title>Ms</Title>
      <Age>25</Age>
      <Address>1 Armageddon</Address>
      <PostCode>SN9 3AA</PostCode>
      <Salary>29000</PostCode>
    </row>
      <action>new</action>
      <Name>J. Smith</Name>
      <Title>Mr</Title>
      <Age>45</Age>
      <Address>45 Jones Drive</Address>
      <PostCode>SN1 3AA</PostCode>
      <Salary>18000</PostCode
  </data>
</steelhead>
```

The Steelhead Application on Department Manager's PC recognises the e-mail, extracts the data and adds the data to an external data file, in this example the e-mail changes one record and adds one record to the existing CSV file on the Department Manager's PC called "Salaries". The file has now been automatically updated and now looks as follows:

Department Manager PC / Filename = Salaries (new)

```
Sender,Name,Title,Age,Address,PostCode,Salary
HR(NY),J. Doe, Mr,33,1 Anywhere,10005,28000
HR (Boston),E. Kennedy,Chappaquidick Drive,12345,350000
HR(London),A. Smith,Ms,25,1 Armageddon,SN9 3AA,29000
HR(London),J. Smith,Mr,45,45 Jones Drive,SN1 3AA,18000
```

This data can now be shared by any application running on the same network as the Department Manager's computer, for example both a spreadsheet calculating the annual cost of all temporary staff and a map plotting the location of all hires. So the effect of the one e-mail is to "update" the data for one or several applications.

b): **Remote switching on and off of pre-programmed features**

The original Program-Flow File is replaced, as a result of receiving an e-mail, with a new Program-Flow File. The following pseudo-code illustrates the process.

Pseudo Code

```
..
LoadProgramFlowFile
..
CheckInBox
For Each Message
  If Message.Sender = "upgrades@steelhead.com"
    If Message.Subject = "Program-Flow Update"
      NewFileData = Decrypt (Message.Body)
      SaveProgramFlowFile (NewFileData)
      LoadProgramFlowFile
    Else
      ..
    End if
  Else
    ..
  End if
Next
```

Example:

The following is a potential Program Flow file on the recipient's PC

Program-Flow File (original)

```
[menus]
Delete = True
Sort   = False
Filter = False

[authorisation]
ExpiryDate = 31/12/2002
CheckLicense = False
EnforceExpiry = False
AllowSending = False

[windows]
AllowMultipleWindows = False
```

An e-mail is received by the recipient updating the Program-Flow

E-mail

To:	user@client.com
From:	upgrades@steelhead.com
Subject:	Program-Flow Update

```
[menus]
Delete = False
Archive = True
Sort   = True
Filter = True

[authorisation]
ExpiryDate = 31/12/2002
CheckLicense = True
EnforceExpiry = True
AllowSending = False

[windows]
AllowMultipleWindows = True
```

The Program-Flow file is updated on the recipient's PC

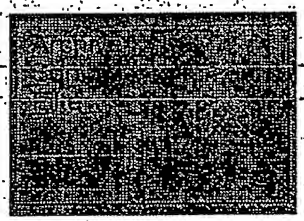
Program-Flow File (new)

```
[menus]
Delete = False
Archive = True
Sort   = True
Filter = True

[authorisation]
ExpiryDate = 31/12/2002
CheckLicense = True
EnforceExpiry = True
AllowSending = False

[windows]
AllowMultipleWindows = True
```

As a consequence, when the Application program is opened the Right Hand mouse click menu would have changed as illustrated, now giving access to new features:

Original Menu

New Menu



c): Remote updating of program code

The original exectuable is replaced, as a result of receiving an e-mail, with a new executable. The following pseudo-code illustrates the process.

Pseudo Code

```
[Shell Program]
If FlagToLoadNewExeOnStartup
  Replace MainProgram with TemporaryExe
End If
Start MainProgram

[Main Program]
CheckInBox
For_Each_Message
  If Message.Sender = "upgrades@steelhead.com"
    If Message.Subject = "Full Update"
      NewFileData = Decrypt (Message.Body)
      Save NewFileData as TemporaryExe
      Set FlagToLoadNewExeOnStartup
    Else
      ..
    End if
  Else
    ..
  End if
Next
```

The e-mail might look like this:

E-mail

To: user@client.com
From: upgrades@steelhead.com
Subject: Full Update

```
Vndiofuynvlsvhn3987593e87nsk
Vnsdhjf577120n3945nbkshh93nb
Djeu348579ene9756bnseoodh93nr
Djndieurytnowiun399379Oudnsk
.........
```

The program is instructed to replace the entire code of the executable with the new code contained within the e-mail

d): **Remote issuing of commands to the applicant's program**

Commands, such as turning on a logfile or sending an error report, can be executed as a result of receiving an e-mail. The following pseudo-code illustrates the process.

Pseudo Code

```
CheckInBox
For Each Message
  If Message.Sender = "commands@steelhead.com"
    CommandData = Decrypt (Message.Body)
    Get Command and Parameters from CommandData
    If Message.Subject = "Steelhead"
      Select Command
        LOGFILE:
          LogFileOutput (param1, param2))
        ERRORREPORT:
          Send E-mail ("errorreports@steelhead.com",
                       Errorreport(param1))
        BACKUPFILES:
          ...
      End Select
    Else
      ...
    End if
  Else
    ...
  End if
Next
```

Example :

A sample e-mail with a command instruction to the Steelhead Application to back-up data and to send an error log back to the applicant (Steelhead).

E-mail

To: user@client.com
From: commands@steelhead.com
Subject: Steelhead

```
BACKUPFILES DATA,temp.dat
TESTDATA temp.dat,errors.log
```

This could also be achieved by a structured e-mail, the structured part being normally encrypted (to prevent tampering):

E-mail

To:	user@client.com
From:	system@steelhead.com
Subject:	Irrelevant Title

This e-mail contains harmless instructions to your Steelhead software. If you can see this message, please goto www.steelhead.com/downloads to download the software.

```
<steelhead>
  <senderlicence>837isn2855ns0d925n30</senderlicence>
  <mailtype>steelhead command</mailtype>
  <data>
    <command>
      <name>BACKUPFILES</name>
      <param1>DATA</param1>
      <param2>temp.dat</param2>
    </command>
    <command>
      <name>TESTDATA</name>
      <param1>temp.dat</param1>
      <param2>errors.log</param2>
    </command>
      </data>
</steelhead>
```

The Steelhead Application recognises the command as intended for itself and executes a back-up and sends an error log.

e): **Remote issuing of commands to other programs**

A command for another program can be contained in an e-mail, either structured or not. The following pseudo-code illustrates the process.

Pseudo Code

```
..
CheckInBox
For Each Message
  If Message.Sender = "commands@steelhead.com"
    CommandData = Decrypt (Message.Body)
    Get Command and Parameters from CommandData
    If Message.Subject = "Steelhead"
      ..
    Else
      Execute (Message.Subject) with Command and Parameters
      ..
    End if
  Else
    ..
  End if
Next
```

Example:

A sample e-mail with a command instruction to another application, in this case a Home Alarm system, to turn on lights in the garage

E-mail

To:	user@client.com
From:	commands@steelhead.com
Subject:	HomeSecurity.exe

```
HOME SECURITY; LIGHTS GARAGE,ON;NOW
ALARM ON 3416
```

This could also be achieved by a structured e-mail, the structured part being normally encrypted (to prevent tampering): Structured example:

E-mail

To:	user@client.com
From:	system@steelhead.com
Subject:	Instructions for HomeSecurity

If you can see this message, please goto www.steelhead.com/downloads to download the software.

```
<steelhead>
  <senderlicence>jf89fjds72938fs09983gnsk</senderlicence>
  <mailtype>external command</mailtype>
  <data>
    <program>
      <name>HomeSecurity.exe</name>
      <command>
        <name>LIGHTS</name>
        <param1 name="location">GARAGE</param1>
        <param2 name="state">ON</param2>
        <param3 name="time">NOW</param3>
      </command>
    </program>
  </data>
</steelhead>
```

The Steelhead Application recognises the command as intended for the Home Security System and passes a command line to the appropriate API to instruct the Home Security System to turn on the garage lights.

Having described particular preferred embodiments of the present invention, it is to be appreciated that the embodiments in question are exemplary only and that variations and modifications such as will occur to those possessed of the appropriate knowledge and skills may be made without departure from the spirit and scope of the invention as set forth in the appended claims. For example, the present embodiments have described the use of XML inserted within an e-mail, however other non XML ways could also be employed such as using a name equals value approach which would be a new way of defining a data structure in text.

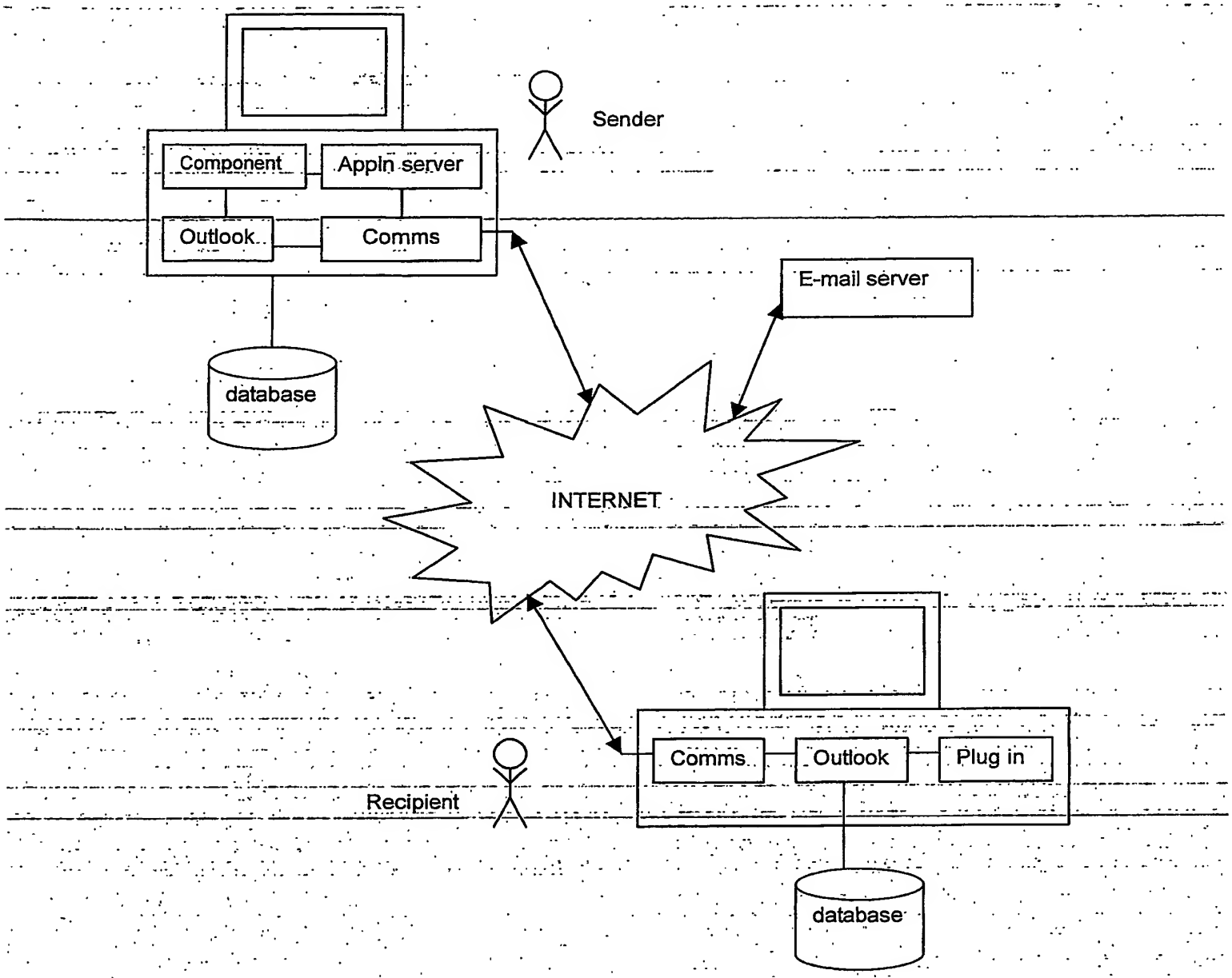

Component
Appln server
Outlook
Comms
database
Sender
E-mail server
INTERNET
Recipient
Comms
Outlook
Plug in
database

FIGURE 1

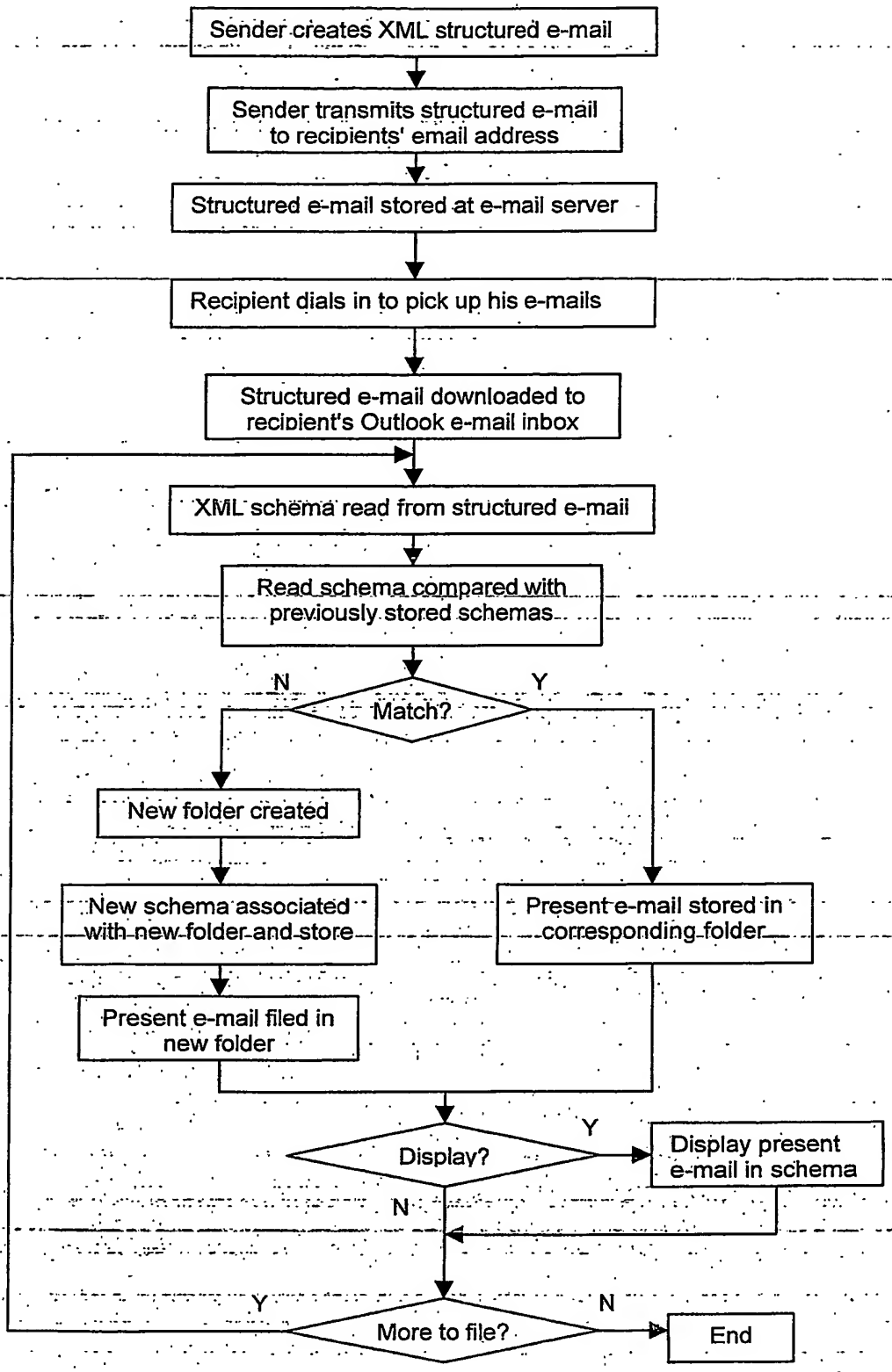

Sender creates XML structured e-mail
Sender transmits structured e-mail to recipients' email address
Structured e-mail stored at e-mail server
Recipient dials in to pick up his e-mails
Structured e-mail downloaded to recipient's Outlook e-mail inbox
XML schema read from structured e-mail
Read schema compared with previously stored schemas
N
Match?
Y
New folder created
New schema associated with new folder and store
Present e-mail filed in new folder
Present e-mail stored in corresponding folder
Display?
Y
Display present e-mail in schema
N
Y
More to file?
N
End

FIGURE 2

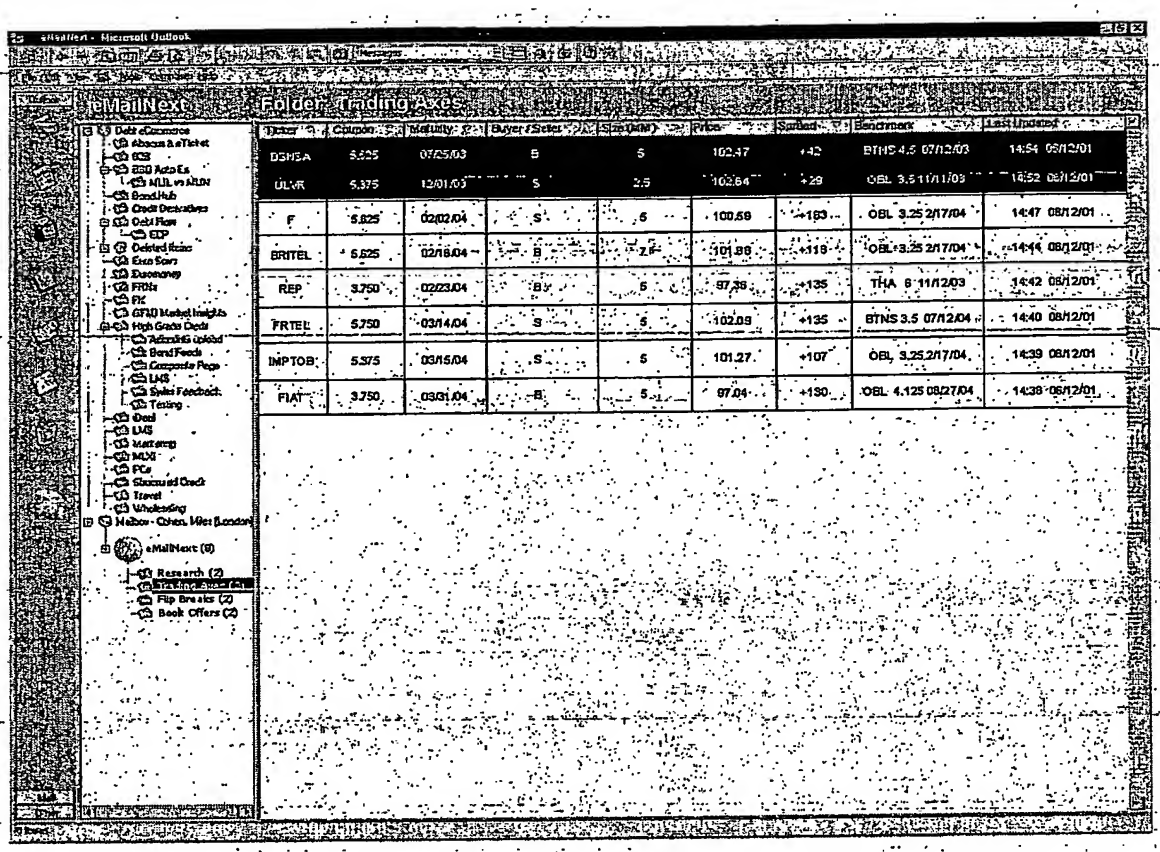


FIGURE 3

```
<SCHEMA>
        <FIELD1 Type ="text">Ticker</FIELD1>
        <FIELD2 Type ="text">Coupon</FIELD2>
etc...
        <FIELD8 Type ="text">Last Updated</FIELD8>
        <TABLE KEY>
                <Ticker>
                <Coupon>
                <Maturity>
etc...
        </TABLE KEY>
</SCHEMA>

<DATA>
        <RECORD1>
                <Ticker>BSNSA</Ticker>
                <Coupon>5.625</Coupon>   etc....
        </RECORD1>

        <RECORD2> etc.....

</DATA>
```

FIGURE 4

NOT TO BE AMENDED

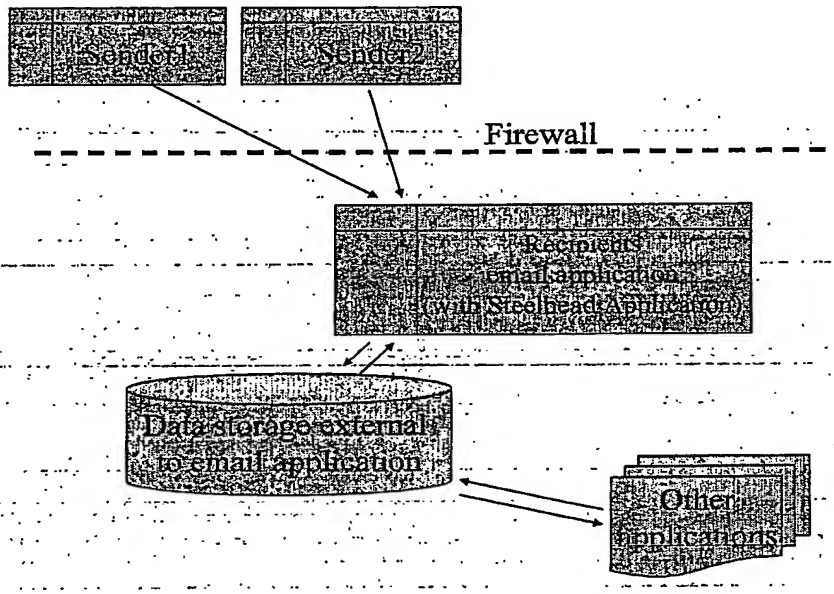

Sender1
Sender2
Firewall
Recipients' email application (with Steelhead Application)
Data storage external to email application
Other applications

FIGURE 5

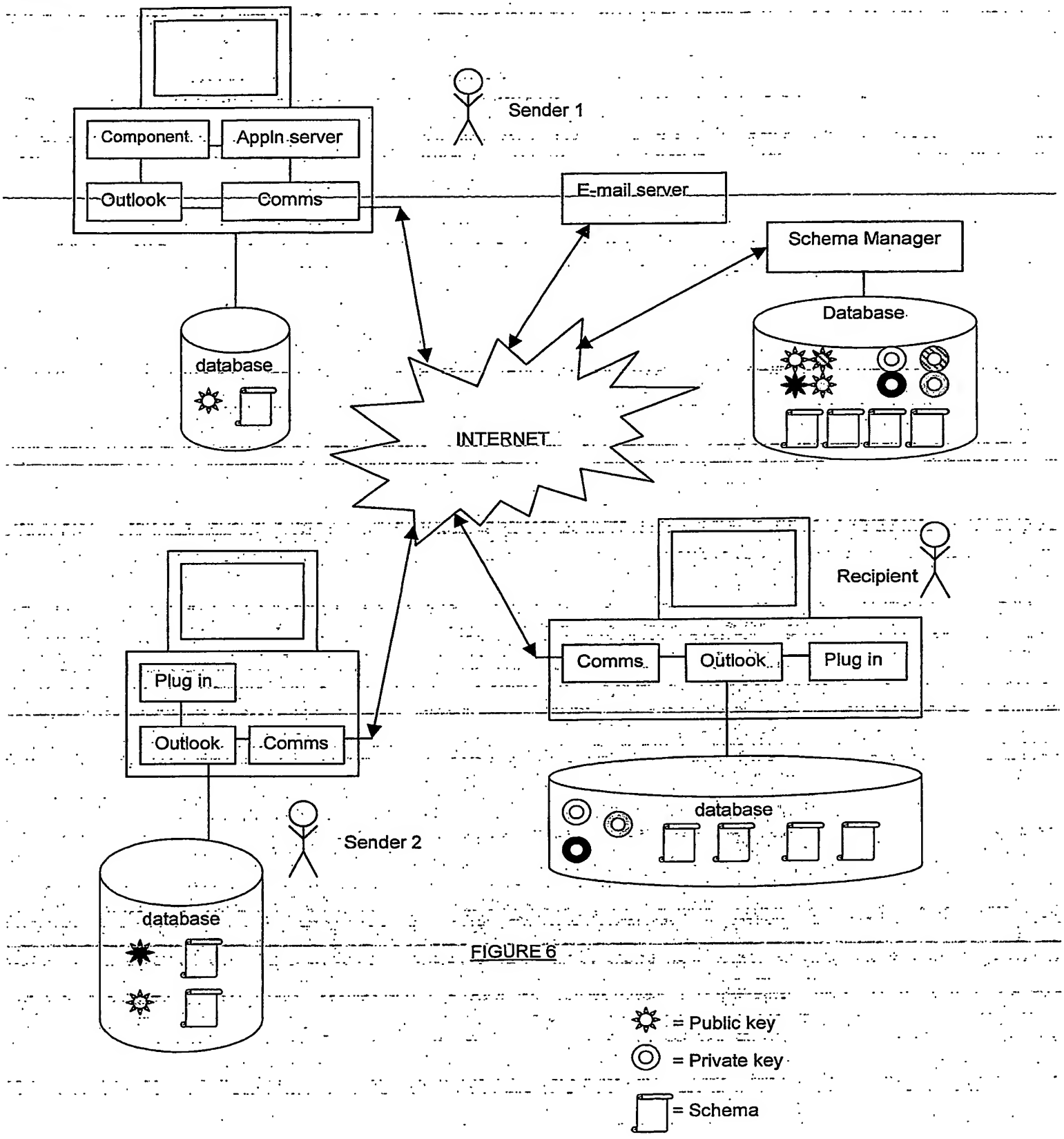

Component
Appln server
Outlook
Comms
Sender 1
E-mail server
Schema Manager
Database
database
INTERNET
Recipient
Plug in
Outlook
Comms
Comms
Outlook
Plug in
database
Sender 2
database
= Public key
= Private key
= Schema

FIGURE 6

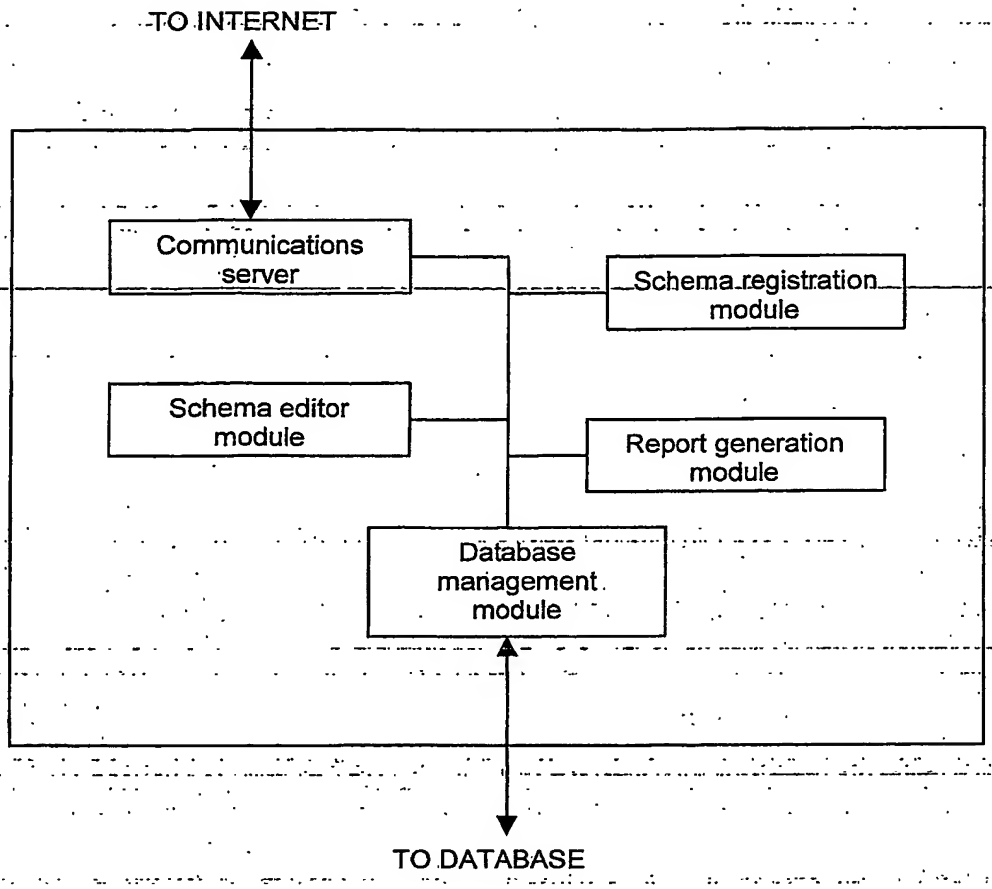

TO INTERNET
Communications server
Schema registration module
Schema editor module
Report generation module
Database management module
TO DATABASE

FIGURE 7

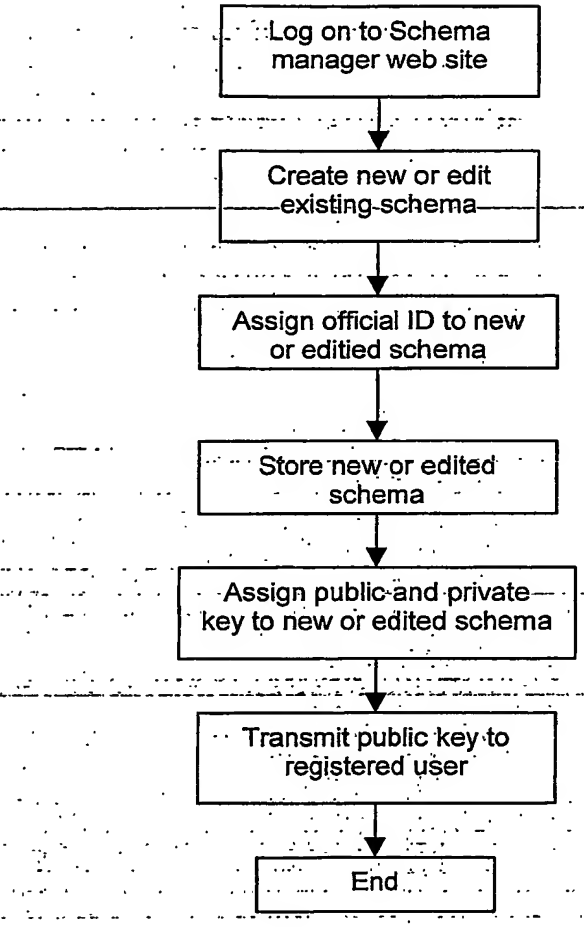

Log on to Schema manager web site
Create new or edit existing schema
Assign official ID to new or editied schema
Store new or edited schema
Assign public and private key to new or edited schema
Transmit public key to registered user
End

FIGURE 8

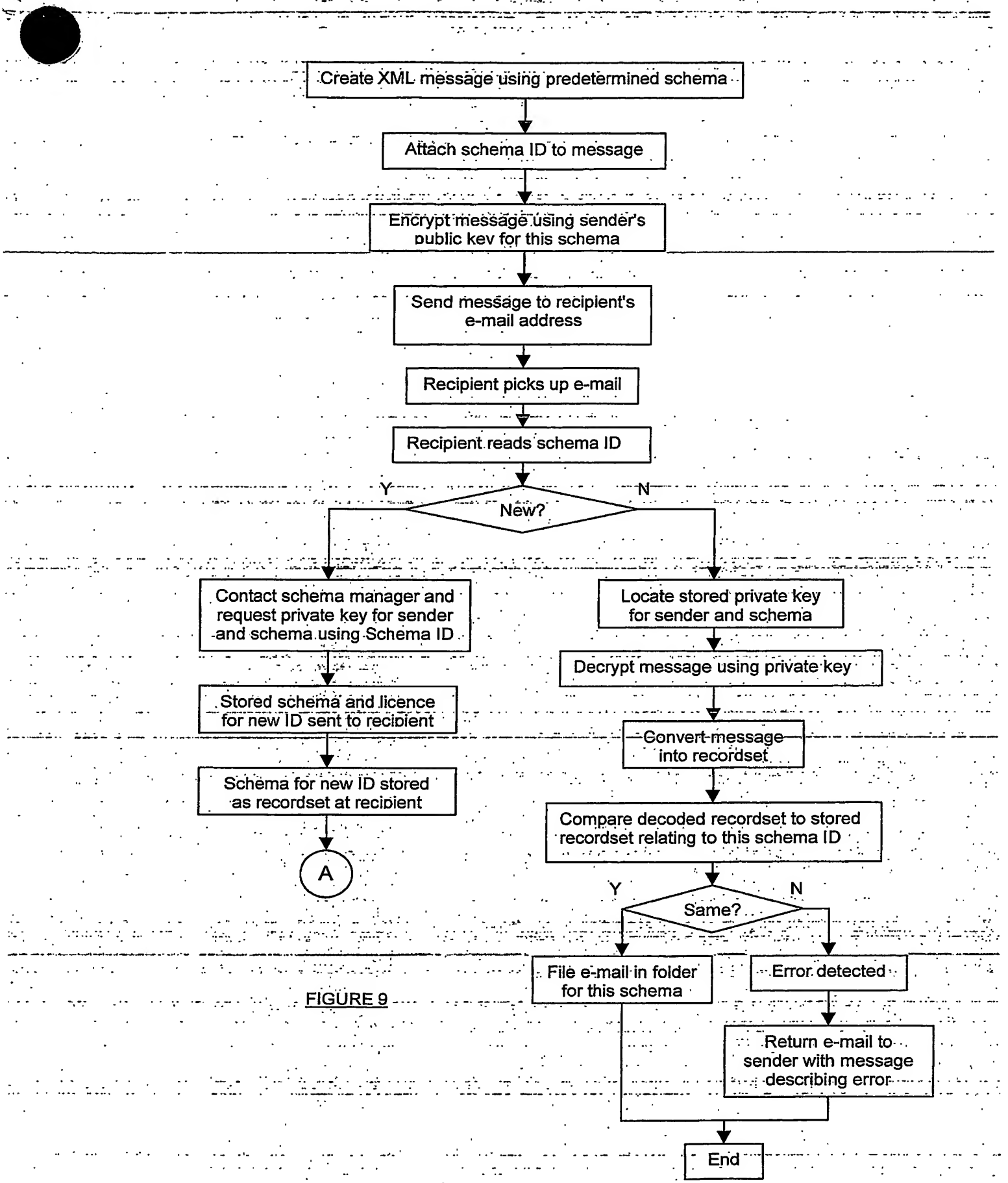

Create XML message using predetermined schema
Attach schema ID to message
Encrypt message using sender's public key for this schema
Send message to recipient's e-mail address
Recipient picks up e-mail
Recipient reads schema ID
New?
Y
N
Contact schema manager and request private key for sender and schema using Schema ID
Stored schema and licence for new ID sent to recipient
Schema for new ID stored as recordset at recipient
A
Locate stored private key for sender and schema
Decrypt message using private key
Convert message into recordset
Compare decoded recordset to stored recordset relating to this schema ID
Same?
Y
N
File e-mail in folder for this schema
Error detected
Return e-mail to sender with message describing error
End

FIGURE 9

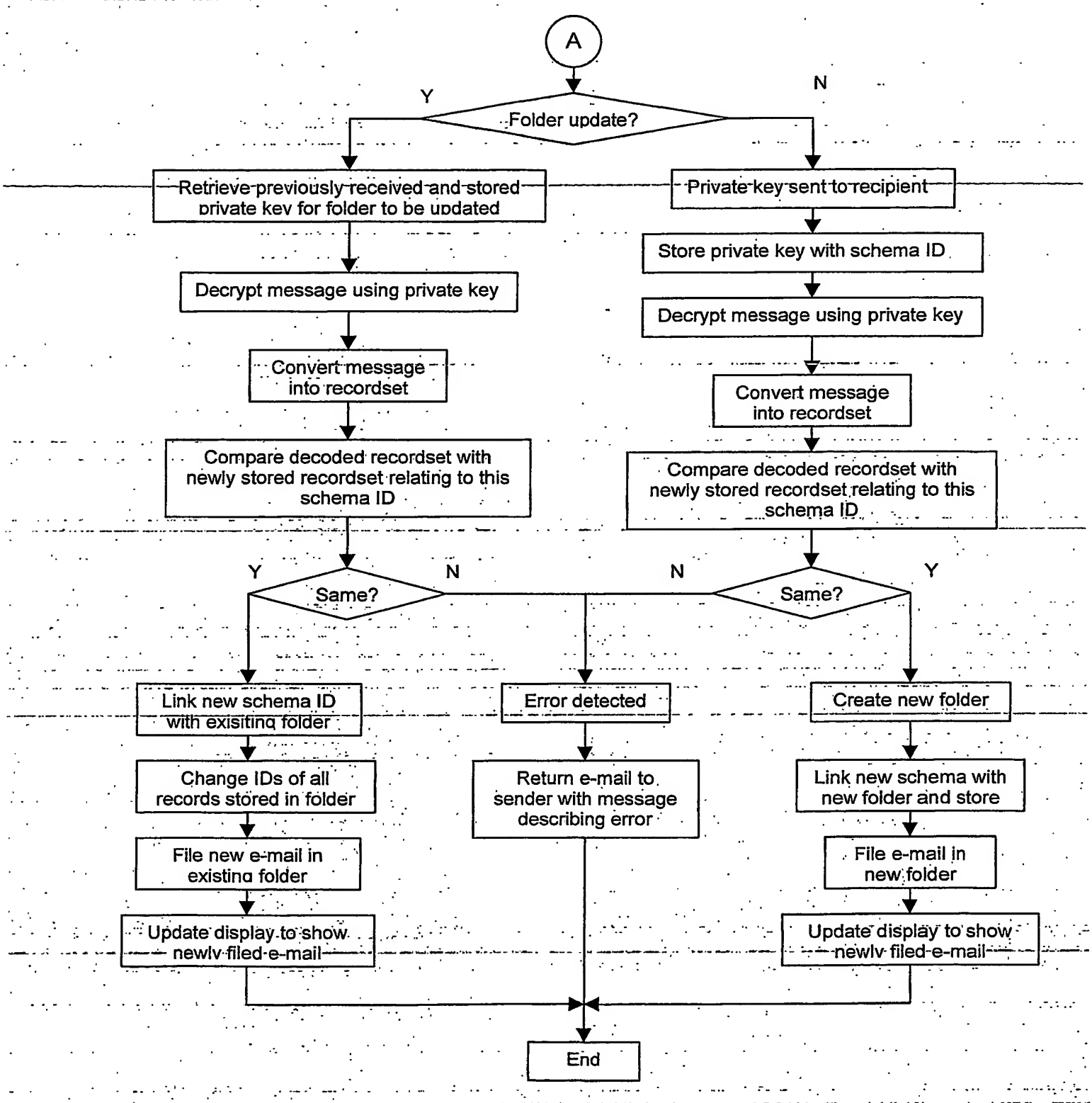

A
Folder update?
Y
N
Retrieve previously received and stored private key for folder to be updated
Decrypt message using private key
Convert message into recordset
Compare decoded recordset with newly stored recordset relating to this schema ID
Same?
Y
N
Link new schema ID with exisitinq folder
Change IDs of all records stored in folder
File new e-mail in existinq folder
Update display to show newlv filed e-mail
Private key sent to recipient
Store private key with schema ID
Decrypt message using private key
Convert message into recordset
Compare decoded recordset with newly stored recordset relating to this schema ID
Same?
N
Y
Error detected
Return e-mail to sender with message describing error
Create new folder
Link new schema with new folder and store
File e-mail in new folder
Update display to show newlv filed e-mail
End

FIGURE 10

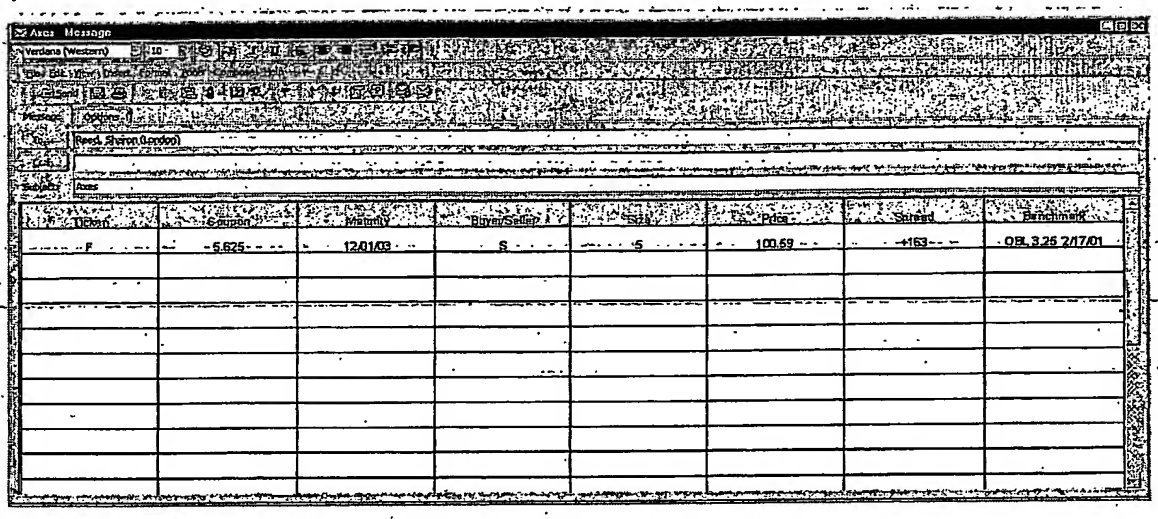


FIGURE 11

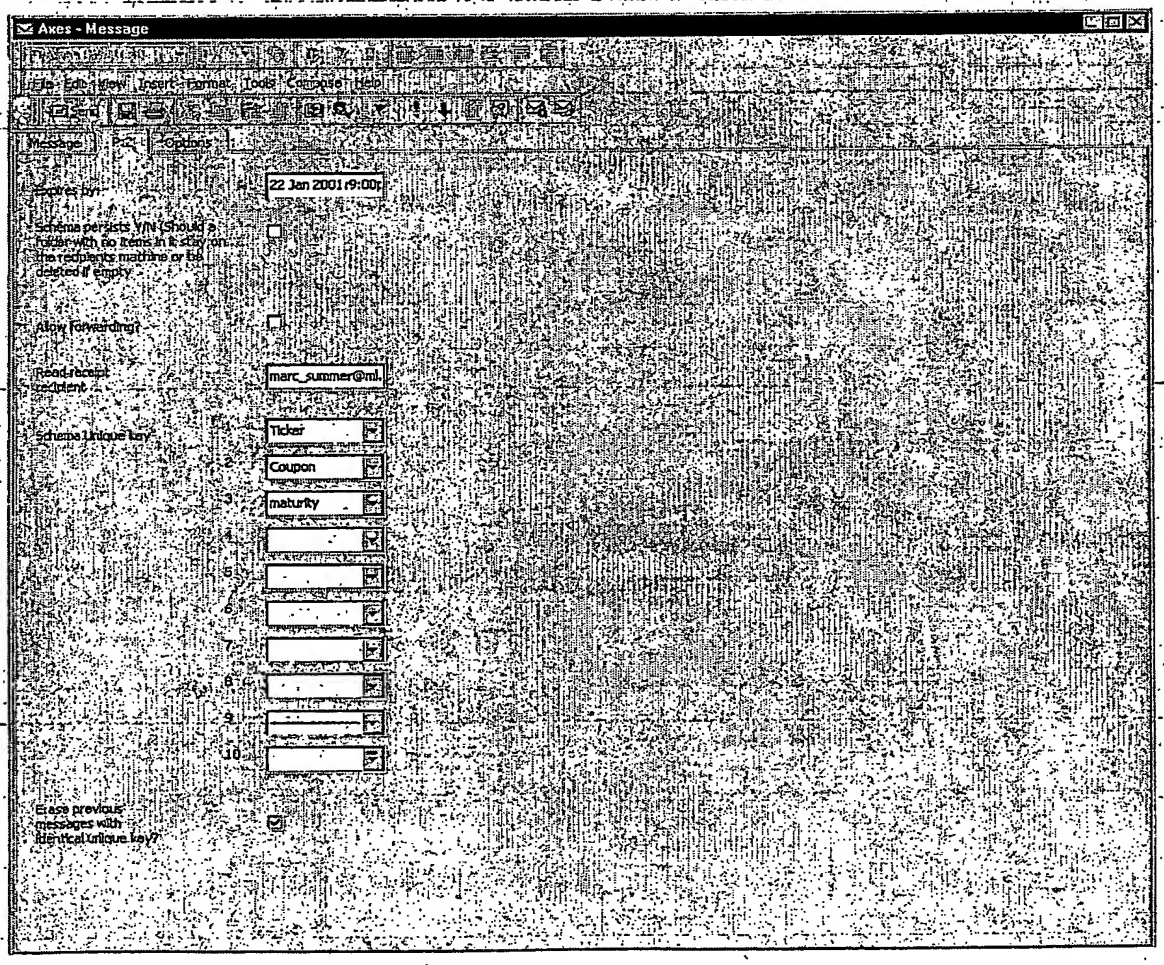


FIGURE 12

NOT TO BE AMENDED

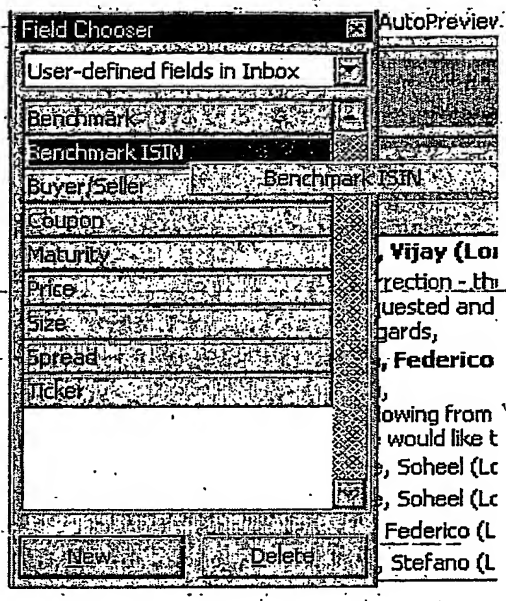


FIGURE 13

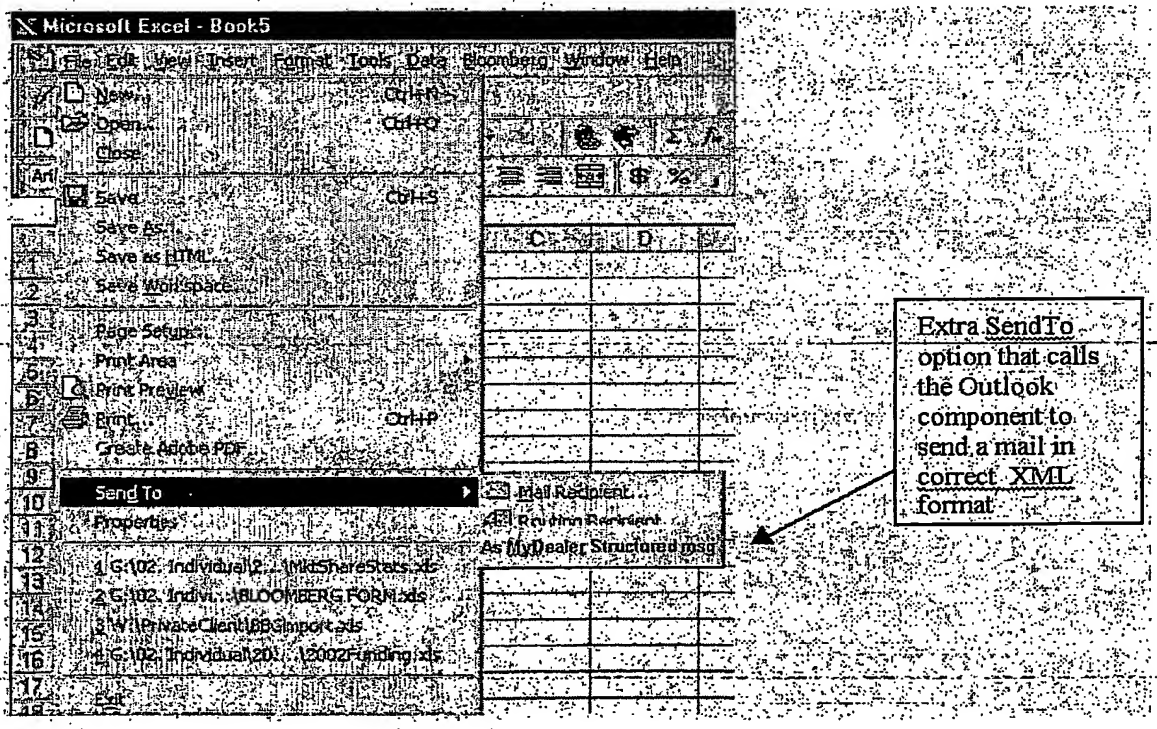


FIGURE 14

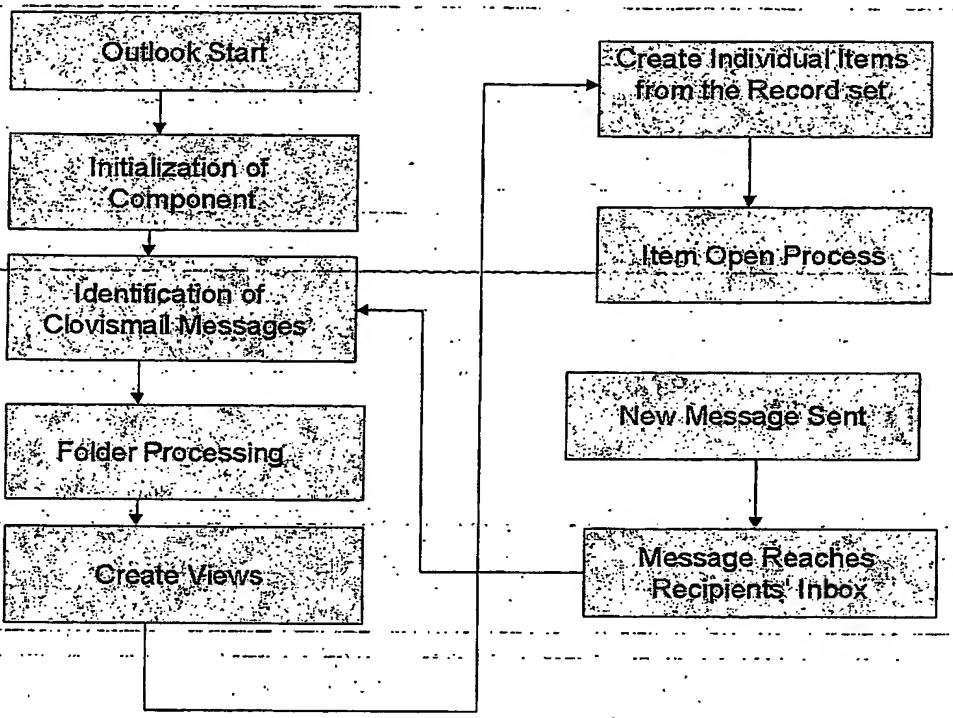


FIGURE 15

Sending Message

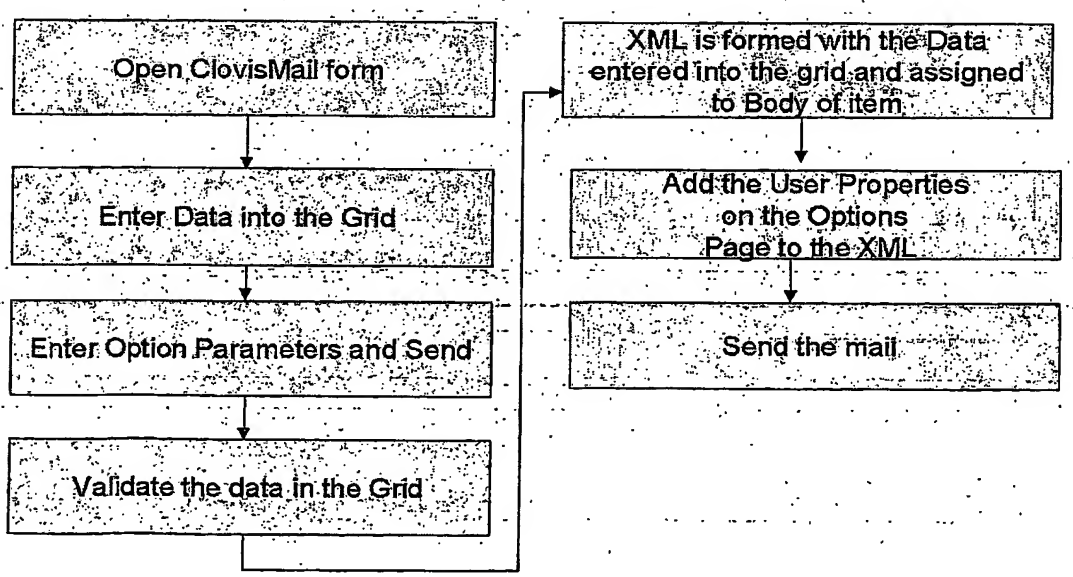


FIGURE 16

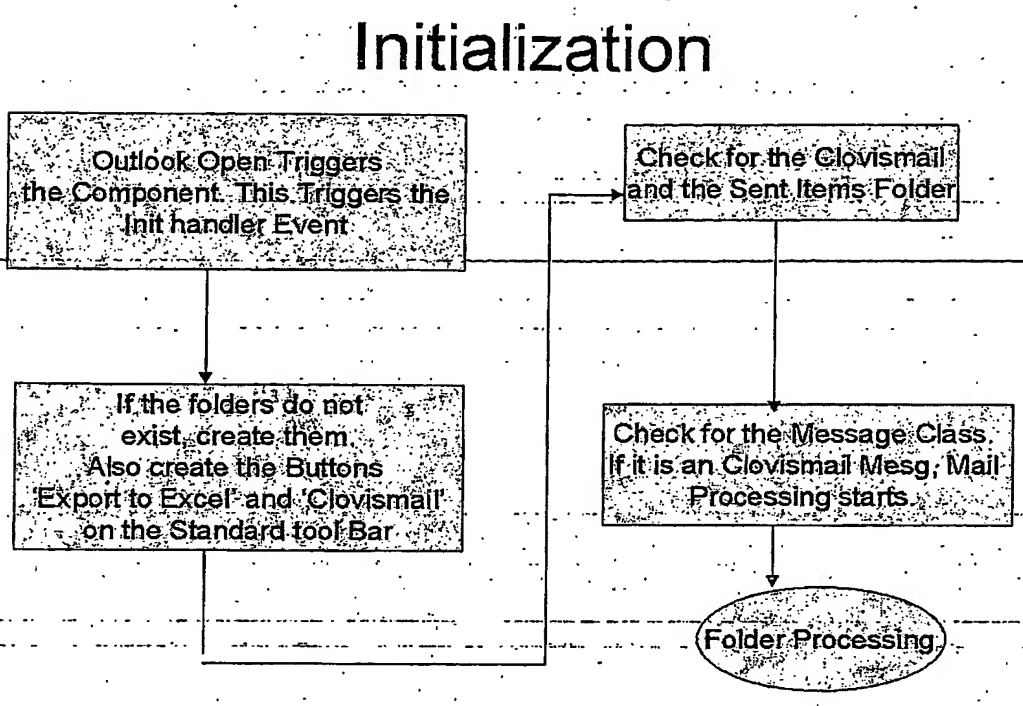


FIGURE 17

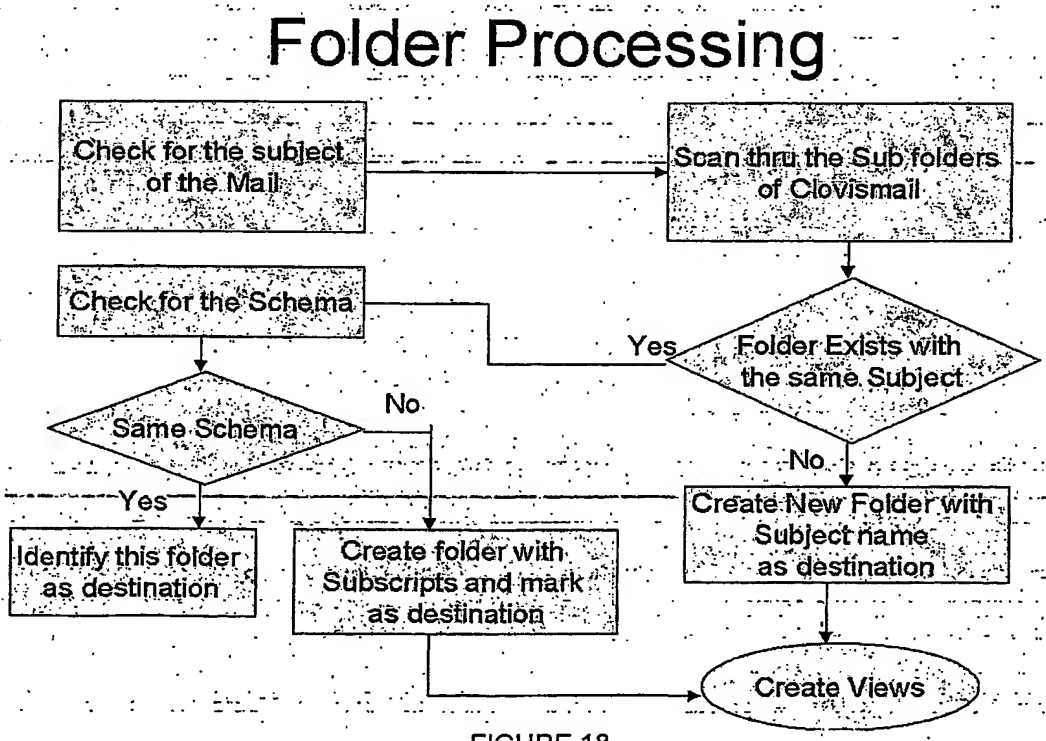


FIGURE 18

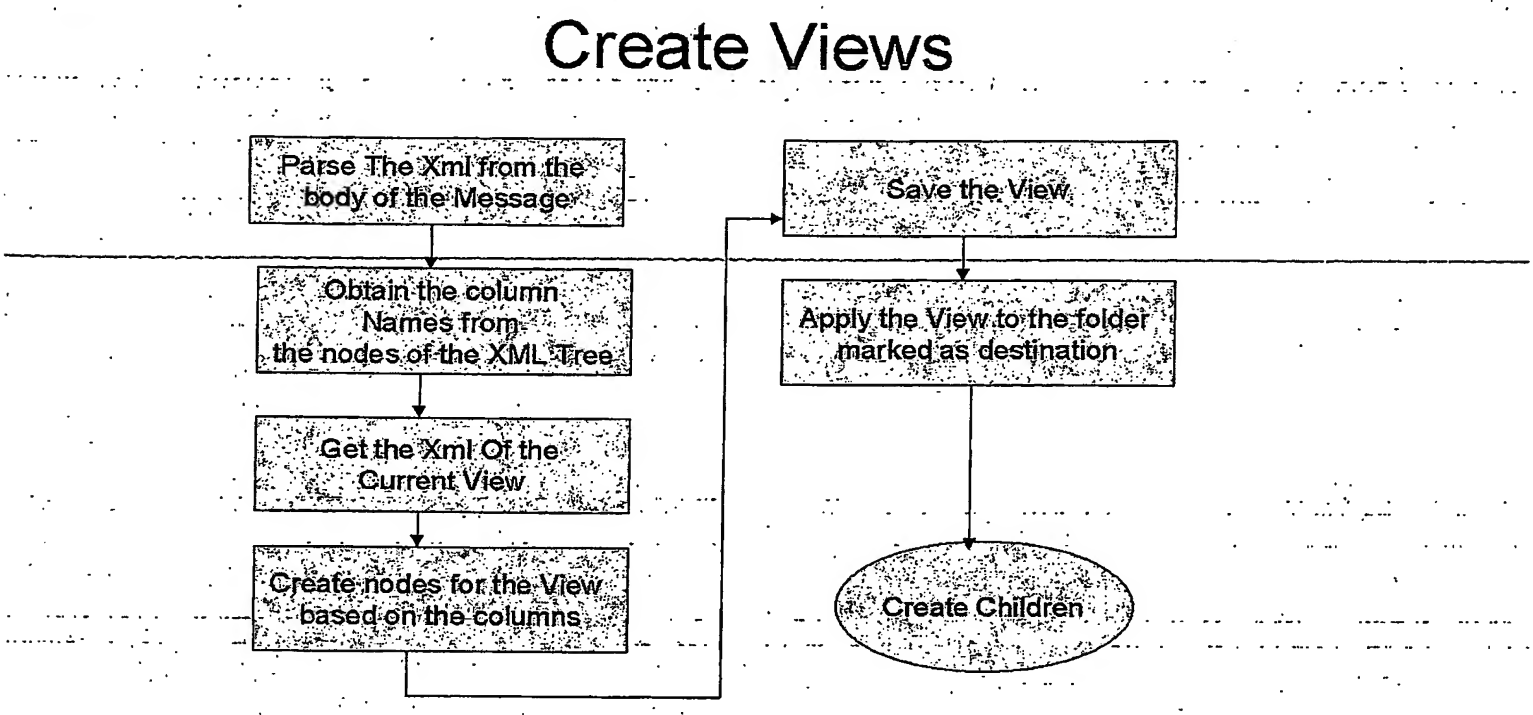


FIGURE 19

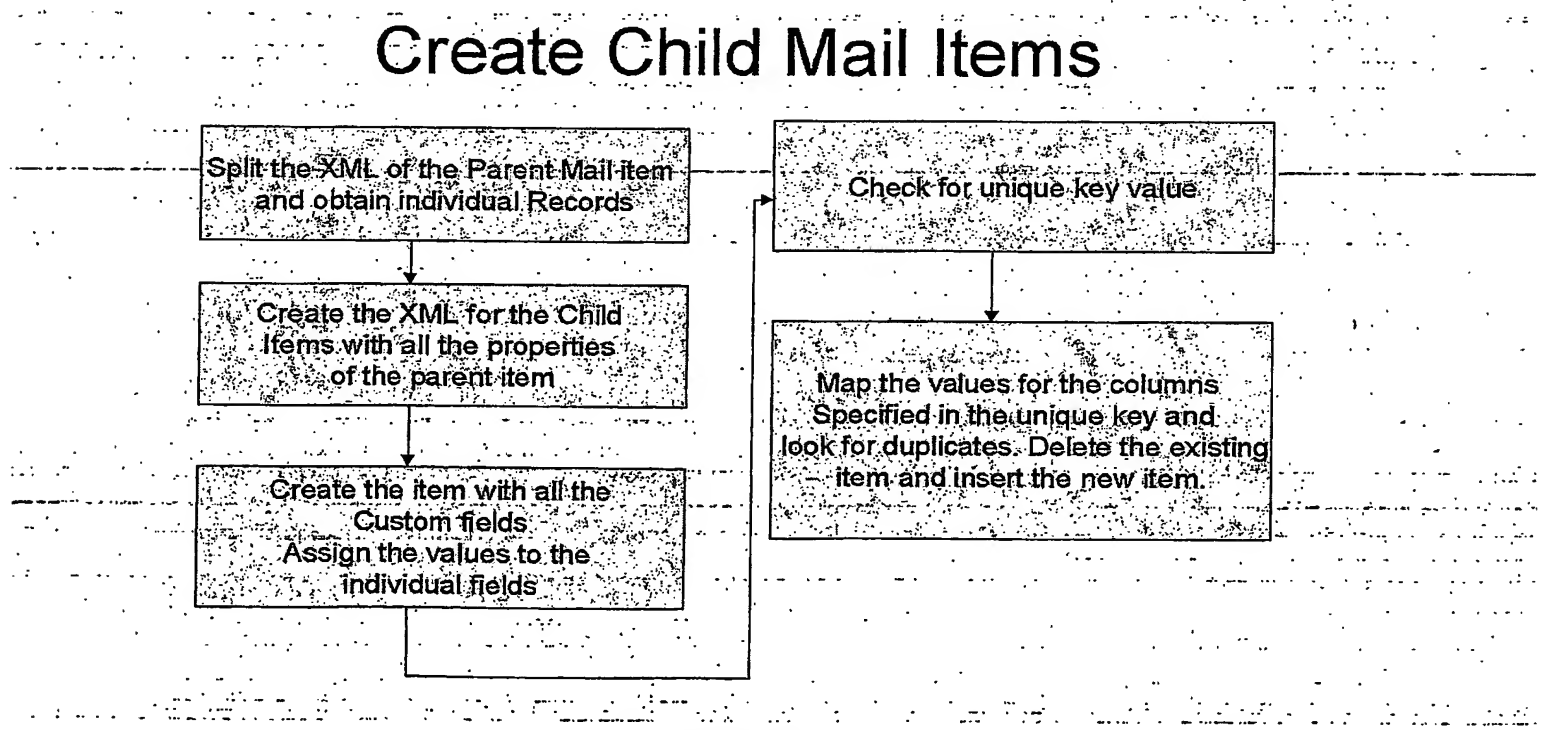


FIGURE 20

Item Processing

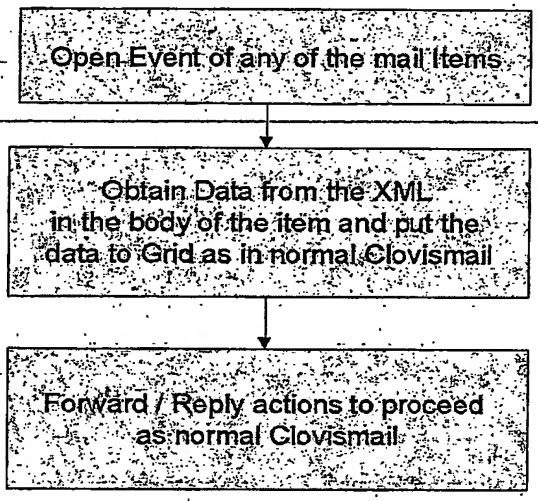


FIGURE 21